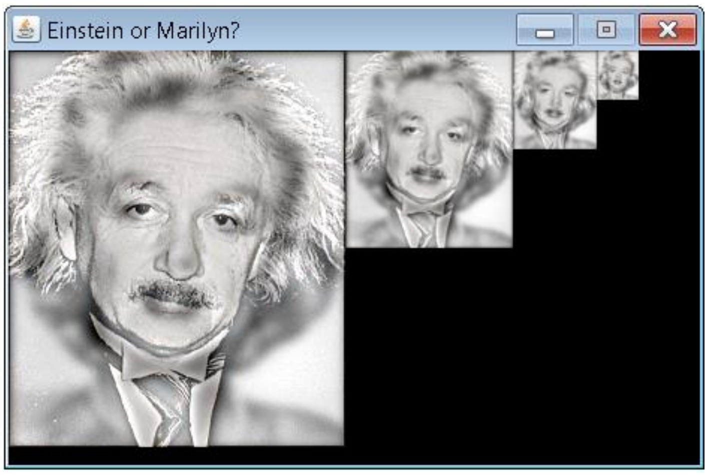
COMP3204 Computer Vision

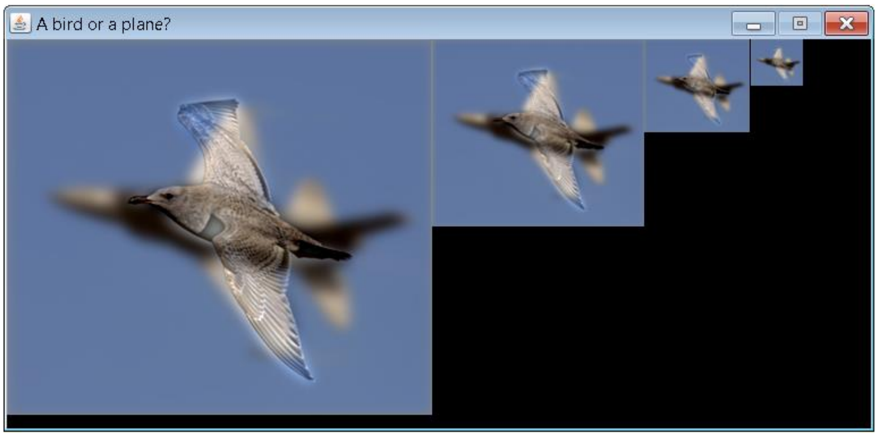
Coursework 2: Image Filtering and Hybrid Images

Results

The goal of this assignment was to write a basic image convolution function and use it to create hybrid images. In this regard, I believe I have succeeded in creating a variety of images which have two different interpretations depending on the viewing distance. Featured in this report are screenshots of the results of the best produced from my program and the given image dataset.



Running the application will open the images, make hybrid images, and demonstrate these hybrids by displaying the image progressively down-sampled to ease the visualisation of a hybrid image, which is demonstrated in the screenshots.



Algorithms

Convolution Algorithm

The template convolution algorithm I have implemented is based on the pseudocode in the book by Mark Nixon and Alberto Aguado *Feature Extraction & Image Processing* in section 3.4.1 page 81. It starts off with a blank black image but fills it in by looping over the pixels in the given image and summing pixel values covered by the kernel.

Below is the Java code of just the template convolution algorithm built in the given MyConvolution class skeleton. The code is very well documented and should be trivial to understand.

```
@Override
        public void processImage(final FImage image) {
                // get image dimensions
                int iRows = image.getRows();
                int iCols = image.getCols();

                // get kernel dimensions
                int tRows = kernel.length;
                int tCols = kernel[0].length;

                // set a temporary image to black
                FImage temp = image.clone().fill(0);

                int trhalf = (int) Math.floor(tRows / 2);
                int tchalf = (int) Math.floor(tCols / 2);

                // loop through all pixels of the original image
                for (int x = 1; x < iCols - 1; x++) {
                        for (int y = 1; y < iRows - 1; y++) {

                                // reset sum to zero
                                float sum = 0;

                                // loop through all points within the kernel
                                for (int iWin = 1; iWin < tRows; iWin++) {
                                        for (int jWin = 1; jWin < tCols; jWin++) {
                                                try {
                                                        // convolve the pixels
                                                        sum += image.pixels[y + jWin - tchalf - 1][x + iWin - trhalf - 1]
* kernel[jWin][iWin];
                                                } catch(ArrayIndexOutOfBoundsException e) { /* ignore */ }
                                        }
                                }

                                // set the new pixel in the temp image with the sum
                                temp.setPixel(x, y, sum);
                        }
                }

                // normalise temp image
                FImage convolved = temp.normalise();

                // return convolved image
                image.internalAssign(convolved);
        }
```

Hybrid Algorithm

The hybrid algorithm is used to construct the hybrid images from two images. It does this by using the template convolution described above to perform a low pass on both images. It then performs a high pass on the second image by subtracting the low pass of the second image from the original image. Finally, it adds the low pass of the first image to the high pass of the second image, resulting in a hybrid image. Success in creating convincing hybrid images relies on the images lining up properly, which thankfully all the given images did.